600507

THE FEASIBILITY OF ADAPTIVE UNSTRUCTURED COMPUTATIONS ON PETAFLOPS SYSTEMS

Rupak Biswas

Numerical Aerospace Simulation Division
MRJ/NASA Ames Research Center

Leonid Oliker (NERSC)
Gerd Heber (Univ of Delaware)
Guang Gao (Univ of Delaware)

WHY MESH ADAPTATION?

- Location of physical features of interest not known in advance
- Location and nature of features, and their interactions may be time dependent
- Limited computational resources in terms of CPU time and memory
- However, complicated logic and DS required
- Need frequent adaptations when solving unsteady problems
- Overhead must be low compared to solver

UNSTRUCTURED GRIDS

- Easier to discretize complex domains
- Easier to locally refine and coarsen
- However, less knowledge base about numerics
- Strategic area of algorithm R&D for future high-end applications

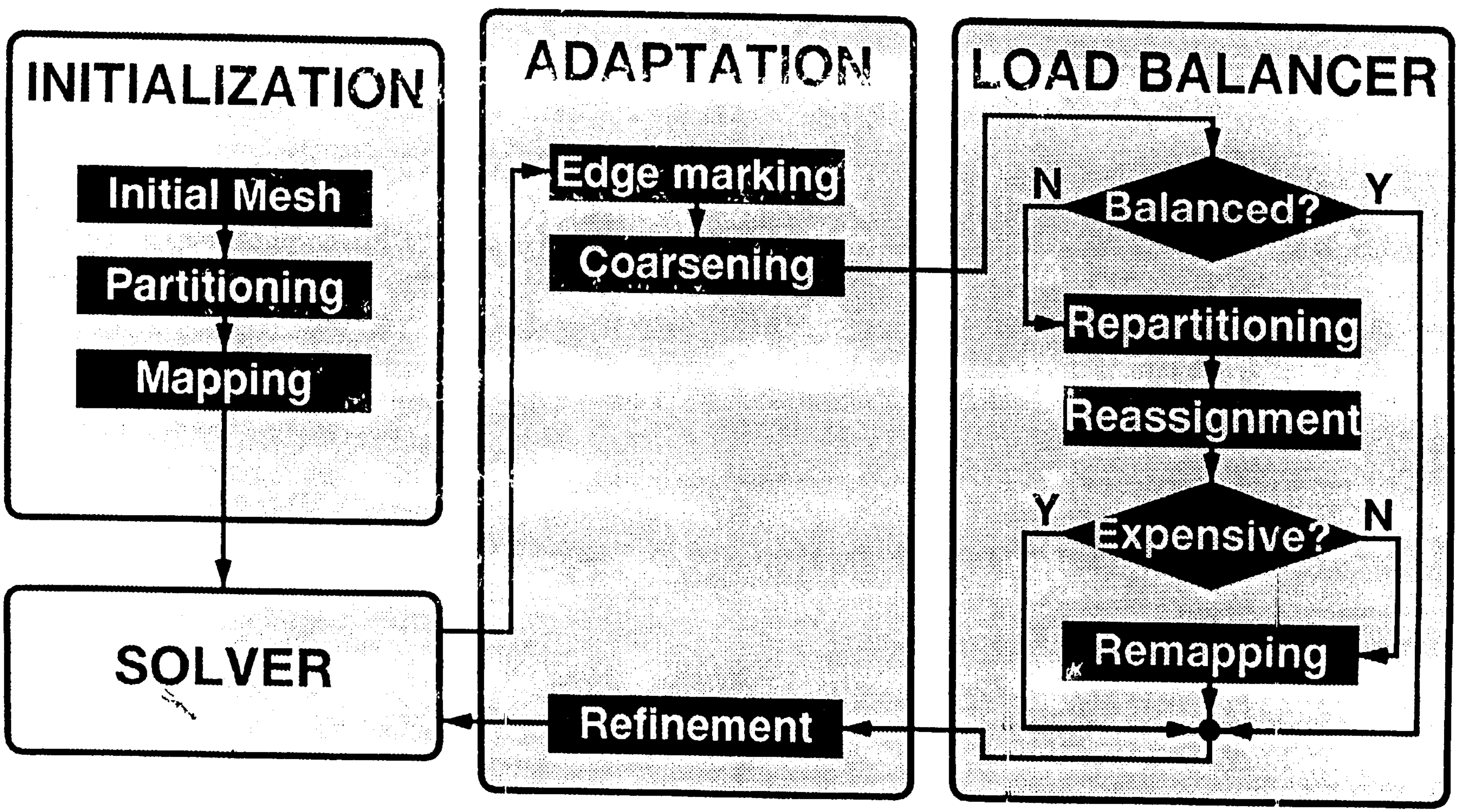
POWER OF MESH ADAPTATION

- Detonation example: rapidly convert chemical energy to heat (lead shock front raises temp, triggering chemical reaction and releasing heat energy)
- Cylindrical rate stick: 10 cm long, 10 cm wide
- Localized reaction zone: ~0.02 mm thick
- At least 10 cells needed across reaction zone
- 1.25G cells for axisymmetric calc on uniform mesh
- 0.25M cells using fine cells in reaction zone
- Resource requirements reduced by factor of 5000

WHY DYNAMIC LOAD BALANCING?

- Local adaptations cause load imbalance among processors
- May also cause significant runtime interprocessor communication
- Require frequent dynamic load balancing when adapting for unsteady problems
- Mesh adaptation and parallelization are at odds

PARALLEL ADAPTIVE COMPUTATIONS



PLUM

- Parallel dynamic load balancing for adaptive unstructured meshes
- Dual graph of initial mesh used for load balancing adapted meshes
- Parallel graph repartitioning
- Several remapping algorithms to assign new partitions to processors
- Efficient data movement scheme (like BSP)
- Metrics to estimate computational gain and communication overhead

DUAL GRAPH OF INITIAL MESH

- Keeps complexity and connectivity constant throughout mesh adaptation
- Load balancing and partitioning times depend only on initial problem size and number of partitions
- Adapted meshes translated to weights for root objects
  - Computational weight
  - Remapping weight
  - Communication weight
- Gives global view of entire computational mesh
- Child elements belong to same partition as parent

SIMILARITY MATRIX CONSTRUCTION

- Matrix S indicates how remapping weights of new partitions distributed over processors
- S_{ij} = sum of remapping weights of all dual graph vertices in new partition j already on processor i

New Partitions

Old Processors	0	1	2	3	4	5	6	7
0		1020		120				
1			500		443	372		
2	129	130		229			43	446
3	13	410	281				198	

PROCESSOR REASSIGNMENT

- Map new partitions to processors such that data redistribution cost minimized
- Cost function usually architecture dependent
- Three different metrics examined:
  - TotalV: total data volume moved among all processors
  - MaxV: maximum data volume to or from processor
  - MaxSR: maximum data volume to and from processor
- Use greedy Heuristic algorithm for quick approximation
- Allow multipartitioning (reduce data volume at the cost of partitioning and reassignment times)

PROCESSOR REASSIGNMENT

- TotalV assumes that reducing network contention and total data volume reduces remapping time
(Solve optimally as MWBG problem in $O(V^2 \log V + VE)$)
- MaxV assumes that reducing data volume for most active processor more important
(Solve optimally as BMCM problem in $O(V^{0.5} E \log V)$)
- MaxSR minimizes sum of heaviest data volume from any processor and to any processor
(Solve optimally as DBMCM problem in $O(V^{0.5} E^2 \log V)$)
- Heuristic algorithm gives suboptimal solution in $O(E)$
(provable bounds on quality)

DIFFUSIVE METHODS

- Global repartitioning generates low edge cuts but may cause high data movement for meshes not changing drastically
- Excess load is diffused to neighboring processors (model the heat equation)
- Neighbors may be determined by hardware topology or domain connectivity
- Usually requires several iterations
- Comes in several versions
- However, communication will still be a primary factor

TYPICAL PETAFLOPS SYSTEM FEATURES

- Large number of processors
  (concurrency, processor-to-processor latency)
- Deep memory hierarchies
  (data locality, processor-to-memory latency)
- Programming paradigm?
  (message passing, shared memory, multithreading)

DATA DECOMPOSITION STRATEGIES

- Graph partitioning
  - Fast multilevel partitioners exist
  - Data locality enforced (but not at the cache level)
  - Works for both distributed and shared memory
- Coloring to form independent sets
  - Requires at least virtual shared memory
  - Prevents race conditions
  - Explicit data remapping not required
  - Easier to program?
  - Need sufficient computational work to mask absence of data locality

SCALABILITY ISSUES

- Communication needs to be low
  (during load balancing (remapping) and runtime (IPC))
- Hide communication under computation
  (asynchronous communication, multithreading)
- Efficient data reuse and data access
  (enhance uniprocessor cache performance)

LATENCY HIDING

- Basically, overlap communication and computation
- Within solver
  (e.g., use a deferred boundary update method)
- Within mesh adaptation
  (e.g., rebuild internal DS while remapping data)
- Use multithreading
  (rapid context switching)

MEMORY HIERARCHY

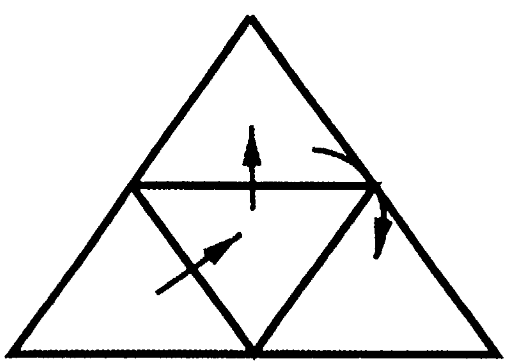
- Crucial to obtain good single processor cache performance
- Difficult for adaptive unstructured applications because of their dynamic and irregular data access patterns
- Use a novel linearization strategy called Self-Avoiding Walks
- Advantage: With fixed problem size and increasing processors, performance improves as data moves up memory hierarchy

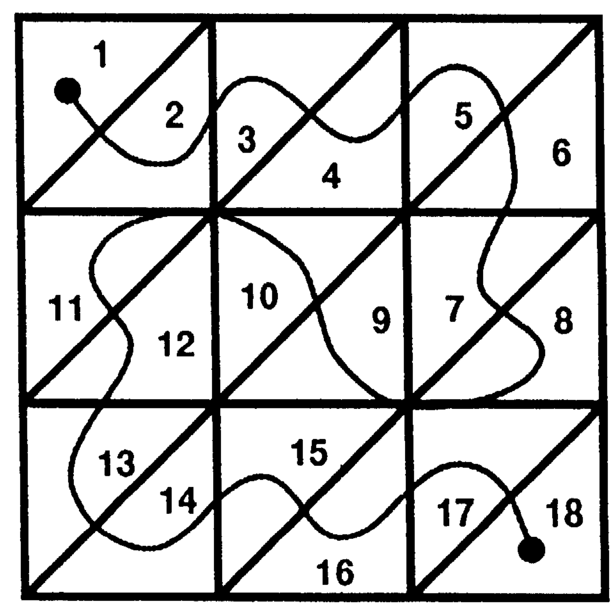
SELF-AVOIDING WALKS

- Serialization of visiting points in higher-dimensional space
- Similar to space-filling curves for structured grids
- But, SAW algorithm is combinatorial (uses mesh connectivity only)
- Easily modified for hierarchical adaptive grids
- Can improve parallel efficiency of irregular grid applications (issues related to locality and load balancing)
- Now in 2D, easily extensible to 3D

PROPER SELF-AVOIDING WALKS

- In 2D, visit each triangle exactly once, entering/exiting over edges and vertices
- In PSAWs, jumping twice over same vertex forbidden

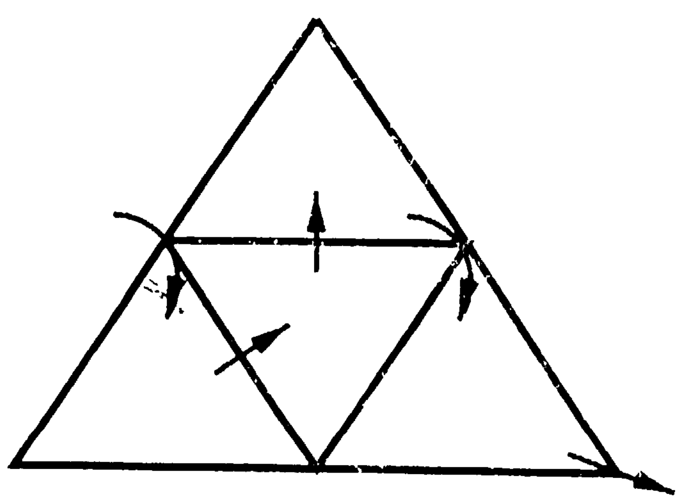


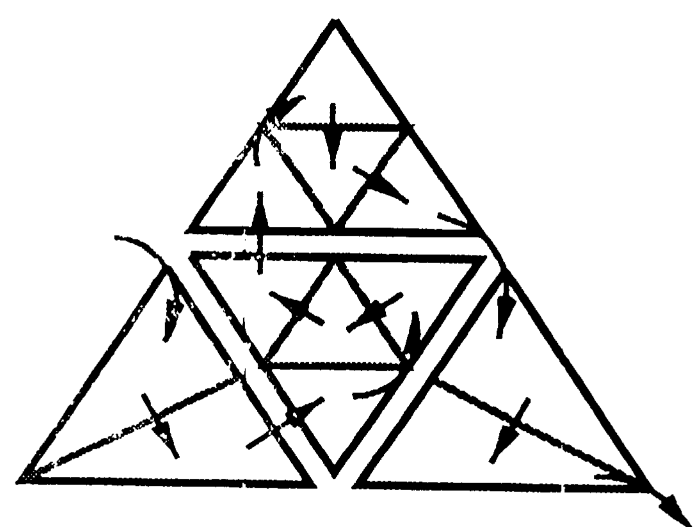


- Existence can be proved for any unstructured mesh (by induction that is also a construction process)

CONSTRAINED PSAW

- Generate PSAW for initial mesh (global)
- Translate footprint to a boundary-value problem
- Exploit regularity of refinement rules to restrict CPSAW to triangle (local)
- Method inherently parallel





CONCLUSIONS

- Need tools and libraries for unstructured mesh adaptation and data remapping
- Overcome challenges in visualization within scope of petaflops computing
- Be able to mask the high communicaton to computation ratio that exists for adaptive unstructured applications
- Performance modeling important, to design next-generation algorithms and distributed systems
- Feasible, provided some significant research challenges can be overcome